966024

Computational Methods for Air Traffic Congestion Delay Optimization

Final Report

Gregory D. Glockner and George L. Nemhauser
Logistics Engineering Center
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

May 20, 1997

[4] G. D. Glockner and G. L. Nemhauser. Dynamic Network Flow with Uncertain Arc Capacities: Formulation and Problem Structure. Technical Report 96-08, Logistics Engineering Center, Georgia Institute of Technology, Atlanta GA, 1996.

[5] G. D. Glockner, G. L. Nemhauser, and C. A. Tovey. Dynamic Network Flow with Uncertain Arc Capacities: Algorithms and Computational Results. Technical report, Logistics Engineering Center, Georgia Institute of Technology, Atlanta GA, 1997.

[6] M. Held, P. Wolfe, and H. P. Crowder. Validation of Subgradient Optimization. *Mathematical Programming* 6, 62–88, 1974.

[7] C. Lemaréchal. Nondifferntiable Optimization. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, eds., *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, pp. 529–572. North-Holland, New York, 1989.

[8] M. Minoux. *Mathematical Programming: Theory and Algorithms.* John Wiley and Sons, New York, 1986.

[9] J. B. Rosen. The Gradient Projection Method for Nonlinear Programming, Part 1: Linear Constraints. *Journal of the Society for Industrial and Applied Mathematics* 8, 181–217, 1960.

[10] S. W. Wallace and T. Helgason. Structural Properties of the Progressive Hedging Algorithm. *Annals of Operations Research* 31, 445–456, 1991.

[11] R. J.-B. Wets. The Aggregation Principle in Scenario Analysis and Stochastic Programming. In S. W. Wallace, ed., *Algorithms and Model Formulations in Mathematical Programming*, pp. 91–113. Springer-Verlag, New York, 1989.

[12] R. J.-B. Wets. Stochastic Programming. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, eds., *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, pp. 573–629. North-Holland, New York, 1989.

	Time			Memory		
Problem	CPLEX	COMET	Ratio	CPLEX	COMET	Ratio
atl6	0:04:28	0:00:37	7.2	225 MB	1 MB	225.0
den3	0:03:29	0:00:21	10.0	210 MB	1 MB	210.0
mco6	0:03:35	0:00:26	8.3	240 MB	1 MB	240.0
sea4	0:03:14	0:00:37	5.2	200 MB	1 MB	200.0
(20,50)	0:07:51	0:00:36	13.1	155 MB	3 MB	51.6
(20,65)	0:12:49	0:00:45	17.1	125 MB	3 MB	41.6
(30,30)	0:13:24	0:01:08	11.8	235 MB	3 MB	78.3
total	0:48:50	0:04:30	10.9	1390 MB	13 MB	106.9

Table 2: CPU Times and Memory Requirements

	LP	COMET					
Problem	optimum	Primal	Diff	% Diff	Dual	Diff	% Diff
atl6	29674.3	29865.6	191.3	0.64%	29534.0	140.3	0.47%
den3	5901.5	5901.5	0.0	0.00%	5899.0	2.5	0.04%
mco6	1284.8	1284.8	0.0	0.00%	1284.7	0.0	0.00%
sea4	47033.4	49642.7	2609.3	5.55%	46229.9	803.5	1.71%
(20,50)	433.7	461.5	27.8	6.40%	423.7	10.0	2.32%
(20,65)	432.8	462.1	29.4	6.79%	424.6	8.1	1.88%
(30,30)	1168.2	1197.3	29.1	2.49%	1145.0	23.2	1.99%
average				3.12%			1.20%

Table 3: Solution Accuracy for COMET

3.5 Publications from This Research

The work under this grant appeared as a technical report [5]. Gregory Glockner's doctoral thesis [3] contains all details for the entire project, including results from prior FAA grants.

References

[1] CPLEX Optimization, Inc., Incline Village NV. *CPLEX*, 4.0 edition, 1995.

[2] G. D. Glockner. Effects of Air Traffic Congestion Delays Under Several Flow Management Policies. *Transportation Research Record* 1517, 29–36, 1996.

[3] G. D. Glockner. *Dynamic Network Flow with Uncertain Arc Capacities.* PhD thesis, Georgia Institute of Technology, Atlanta GA, 1997.

use when compared with a commercial LP solver. More importantly, COMET finds good solutions to problems that are too large to be solved by commercial LP software.

Seven flow management test problems are described in Table 1. Complete results may be found in [5]. In the table, Scen represents the number of scenarios, Comm represents the number of

Problem	Nodes	Arcs	Scen	Comm	Row	Col	Non-0
atl6	565	1078	6	76	353002	491568	1189400
den3	878	1640	3	110	356987	541200	251790
mco6	707	1304	6	66	352788	516384	1196448
sea4	859	1605	4	73	291600	468660	1031928
(20,50)	80	139	50	45	255000	312750	807750
(20,65)	80	139	65	45	334335	406575	1055745
(30,30)	130	229	30	75	355800	515250	1211850

Table 1: Test Problems

commodities, and Nodes and Arcs represent the number of nodes and arcs in the graph $\mathcal{G}$. Row, Col, and Non-0 specify the rows, columns, and nonzeros in the LP matrix.

Each problem was tested with CPLEX's dual simplex method [1] and COMET. Table 2 summarizes both the CPU times and the memory use. The programs were tested on an RS/6000 model 590. Table 2 contains a lower bound for CPLEX's memory use and an upper bound for COMET's memory use. COMET saves about an order of magnitude in CPU time and about two orders of magnitude in memory use. The lower memory requirements result in a better "wall-clock" time for COMET since the operating system does less paging of virtual memory. They also suggest that only COMET can solve these problems within the standard memory configurations of a desktop PC.

The solution accuracy for COMET is found in Table 3. COMET's primal solutions are generated by the primal heuristic, which causes the primal gap. Since the dual optimization is an iterative procedure, we could improve the dual solutions by increasing the number of iterations. However, this demonstrates that compath decomposition finds nearly optimal primal and dual solutions using far less memory and time than CPLEX takes to find an optimal LP solution.

primal heuristic greedily constructs a solution by augmenting the existing flow as much as possible along the cheapest feasible compath. There can be at most $O(K|\mathcal{A}|+1)$ augmentations to this heuristic. Since we can find a cheapest compath in time $O(K|\mathcal{A}|)$, it follows that the heuristic takes $O(K^2|\mathcal{A}|^2)$ time.

The Lagrangian function (2) is piecewise linear and concave. Thus, we use nonsmooth optimization techniques to optimize the Lagrangian dual (3). Several nonsmooth optimization algorithms are described in [7]. In this project, we developed a new direction that approximates the direction generated by bundle methods. Given a set $G = \{g^1, \ldots, g^h\} \subseteq \partial \ell(\pi)$ of supergradients, we construct the search direction

$$d = \frac{\sum_{g \in G} g/\|g\|^2}{\sum_{g \in G} 1/\|g\|^2}. \tag{5}$$

The direction (5) is successful with the compath master problem, though we have not tested its effectiveness with arbitrary nonsmooth optimization problems.

Given a search direction, we must also generate an appropriate step size α and ensure that the new iterate $\pi' = \pi + \alpha d$ is feasible. We use the rule developed in [6] and described in [7, 8]. Specifically, let

$$\alpha \leftarrow \beta \frac{\ell(\pi) + \bar{z}}{\|d\|^2}, \tag{6}$$

where $\beta \in (0, 2)$ is a constant and $\bar{z}$ is the unknown optimum value. To ensure that π' is feasible, we must require that $\pi' \leq 0$. We use a projection method based on Rosen's gradient projection method [9]. We obtain a feasible dual solution π' by projecting the direction and the iterate. The combination of these two projections reduces the bad effects of being near the boundary of $\pi \leq 0$.

3.4 Computational Testing

A real-world flow management problem can be very large. To be practical, we need to be able to solve an instance of this problem quickly. In this part of the project, we test the running time of our algorithms against general-purpose optimization software.

In [5], we describe an implementation of compath decomposition called COMET and present computational results for multicommodity and single commodity problems. COMET generates a nearly optimal solution to the Lagrangian dual resulting from compath decomposition. A heuristic generates primal solutions, and marginal values from the heuristic are used to obtain an initial dual solution. In solving the linear program (1), COMET significantly reduces CPU time and memory

Our algorithm for finding a cheapest compath is based on dynamic programming. First, we define a partition of the scenarios similar to the approach in [11]. For each time t, let Ω_t be the coarsest partition of the scenarios Ω such that if $B \in \Omega_t$ and $k, k' \in B$, then $t \leq \tau(k, k')$. In other words, each B is a maximal subset of Ω such that all scenarios in B are indistinguishable at time t. The sets $B \in \Omega_t$ are known as *scenario bundles* [12] and can be represented as nodes in a *scenario tree*. Likewise, each scenario bundle $B \in \Omega_t$ can be partitioned at time $t' > t$. The collection of scenario bundles that result from partitioning B at time t' are denoted by $B_{t'}$.

Define $f(i, B)$ to be the cost of a cheapest compath from node i to the sink n over the scenario bundle $B \in \Omega_{t(i)}$. If no path exists from i to the sink n, then we say that $f(i, B) = \infty$ for all $B \in \Omega_{t(i)}$. Since the scenarios in B are indistinguishable at time $t(i)$, we must select a single arc (i, j) from node i. This gives the recursion

$$f(i,B) = \min_j \left\{ \sum_{k \in B} p^k c_{ij} - \pi_{ij}^k + \sum_{B' \in B_{t(j)}} f(j, B') \right\} \tag{4}$$

with boundary conditions $f(n, B) = 0$ for all $B \in \Omega_{t(n)}$. Thus, the cost of a cheapest compath from 1 to n over the scenarios Ω is $f(1, \Omega) = z_c$. For each pair (i, B), the cheapest compath recursion finds an optimum arc (i, j) to traverse.

By ordering all nodes such that $t(j) \geq t(i)$ for all $j > i$, we can solve the recursion sequentially from n down to the source, $i = 1$. This recursion only needs to scan each arc $a = (i, j)$ when its starting node i is reached. Thus, each arc is scanned exactly once for each scenario bundle $B \in \Omega_{t(i)}$. Hence, this algorithm finds the cheapest compath in time $O(K|\mathcal{A}|)$. The running time of this algorithm is polynomial in terms of the length of the input data, which consists of the graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and the capacity scenarios $\{u^1, u^2, \ldots, u^K\}$. In [4], we argue that this is the fastest possible algorithm for finding a cheapest compath.

3.3 Master Problem Algorithm

The master problem algorithm uses a primal heuristic and Lagrangian optimization to generate a nearly optimal primal integral solution and an optimum dual solution. Marginal values from the primal heuristic give an initial dual solution. Then, the primal and dual solutions are updated alternately. Better dual solutions improve the cost vector used to generate a primal solution, and better primal bounds improve the step size used by the dual optimization.

From the compath decomposition theorem, any solution x to (1) can be decomposed by compaths. Reversing this process, we can build a solution by assigning flows along compaths. The

In (1), (1.1) are the flow balance constraints, (1.2) are the capacity constraints, (1.3) are the nonanticipativity constraints, and (1.4) are the nonnegativity constraints. The nonanticipativity constraints ensure that a decision cannot anticipate which scenario may occur when some scenarios are indistinguishable. Specifically, if $t(a) \le \tau(k,k')$ for some arc a, then scenarios k and k' are identical up to time $t(a)$, and so the flow on a under scenarios k and k' must be identical. The size of (1) can be reduced by eliminating redundant capacity constraints (1.2). Also, the number of nonanticipativity constraints can be reduced from $O(K^2|\mathcal{A}|)$ to $O(K|\mathcal{A}|)$ by sorting the scenarios according to [10].

Our decomposition scheme places the capacity constraints (1.2) in the master problem and all other constraints in the subproblem. This gives the Lagrangian function

$$\begin{aligned} L(\pi) \;=\; \sum_k \pi^k u^k \;+\; \min \sum_k (p_k c - \pi^k)x^k & \\ N x^k \;=\; b \quad \forall k & \\ x_a^k - x_a^{k'} \;=\; 0 \quad \forall a \in \mathcal{A}; \forall k,k' : t(a) \le \tau(k,k') & \\ x^k \;\ge\; 0 \quad \forall k & \end{aligned} \tag{2}$$

and Lagrangian dual

$$z^* \;=\; \max_{\pi \le 0} L(\pi^1, \ldots, \pi^K). \tag{3}$$

To evaluate the Lagrangian function, we must solve an optimization subproblem. The solutions to this optimization subproblem are very special. To motivate the concept, consider a single scenario acyclic network flow problem with a single source and a single sink. If we relax the arc capacities, the optimum solution allocates all flow to the cheapest source-sink path. We generalized this result to the multiple scenario dynamic network flow problem. We say that the source-sink paths q, q' are compatible for scenarios k, k' if the paths are identical up to time $\tau(k,k')$. Thus, a set of paths $\{q^1, \ldots, q^K\}$ is a compath if each pair q^{k_1}, q^{k_2} is compatible for the corresponding scenarios k_1, k_2. We proved in [4] that the solutions to the subproblems in (2) are flows that correspond to compaths.

3.2 Subproblem Algorithm

By itself, this compath theorem is not helpful in solving traffic flow management problems. We also developed an algorithm for finding a cheapest compath. Thus, we can solve the subproblem in the Lagrangian function (2) by finding a cheapest compath with respect to the costs $(p^1c - \pi^1, \ldots, p^Kc - \pi^k)$ and placing all source-sink flow along the arcs in the compath. In traffic flow management, we may think of a compath as a particular flight plan that is contingent on uncertain weather.

2 Work Summary

The model development and testing was completed under prior FAA support. We describe these results in [2]. In this project, we apply linear programming decomposition schemes for the dynamic network flow model. The research for this project may be divided into four sections: decomposition structure, subproblem algorithm, master problem algorithm, and computational testing. Decomposition structure covers the special structure in the dynamic network flow problem. We then give an algorithm for solving the subproblems that result from compath decomposition. To obtain a complete solution, we use a master problem algorithm. Finally, we test the effectiveness of our decomposition scheme versus general-purpose optimization software.

In this project, we completed the research for these four sections.

3 Summary of Results

The model is described in [2]. For a complete description of the decomposition structure and subproblem algorithms, see [4]. We summarize these results here.

3.1 Formulation and Decomposition Structure

We represent a flow management model with a time-space network. Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be the directed graph where each node $i \in \mathcal{N}$ represents a location at a particular point in time $t(i)$. The time $t(a)$ of an arc $a = (i, j)$ equals $t(i)$, the time of the initial node i. The flow represents the flights, and the arc capacities represent the capacity restrictions on runways or airspace. We define a scenario as one set of arc capacities, and we let $k \in \Omega = \{1, \ldots, K\}$ be the indices of the scenarios. Let p^k be the probability weight for scenario k, and let N be the network flow matrix. Let u^k be the vector of capacities for scenario k, and let x^k be the vector of flows for scenario k. Let $\tau(k, k')$ be the latest time that scenarios k and k' are identical. We formulate the dynamic network flow problem as

$$\begin{aligned}
z^* \;=\; \min \;\; & \sum_k p_k(cx^k) & & \\
Nx^k &= b & \forall k & \qquad (1.1) \\
x^k &\le u^k & \forall k & \qquad (1.2) \\
x_a^k - x_a^{k'} &= 0 & \forall a \in \mathcal{A}; \forall k, k' : t(a) \le \tau(k, k') & \qquad (1.3) \\
x^k &\ge 0 & \forall k. & \qquad (1.4)
\end{aligned} \qquad (1)$$

1 Project Summary

Under a prior FAA grant, we developed a dynamic network flow model for central flow (ATCSCC). This is a robust model that uses uncertain future capacity scenarios to generate optimal flow management solutions. We used a simulation to compare the performance of this model versus other central flow models. This simulation showed that our dynamic network flow model has potential to reduce expected delay costs by several percent.

Because the model incorporates multiple capacity scenarios, a real-world problem can be very large. The purpose of this project is to develop fast algorithms for generating a flow management solution from the dynamic network flow model. We use a software prototype to test the effectiveness of these algorithms.